1. Motivation
2. Project Overview
3. Algorithm Development
    1. Database
    2. Preprocessing
    3. Linear Predictive Coding
    4. Feature Extraction
    5. Neural Networks
4. Results
5. Future Directions
6. The Team

Motivation

Purpose

Human speech contains a remarkable amount of information: semantic meaning, geographical proclivities, and emotional delivery. Individuals who experience difficulty reading emotion from spoken word suffer from impaired social interaction. Automating this process would greatly enhance the ability of such individuals to engage socially. We sought to isolate emotional content by designing a classification algorithm to reliably select among 15 common emotional states independent of semantic significance. This task challenges the precision with which an emotion can be identified; even normally functioning individuals only marginally exceed chance. Linear predictive coding was executed on a labeled, semantically neutral database and statistically characterized to generate a feature vector for each sample. These were used to train a multiclass neural network. Performance was evaluated on a distinct subset of the database.

Useful References

1. D. Ververidis, C. Kotropoulos, and I. Pitas. "Automatic Emotional Speech Classification." ICASSP, 2004.
2. H. Sato, Y. Mitsukura, M. Fukumi, and N. Akamatsu. "Emotional Speech Classification with Prosodic Parameters by Using Neural Networks." Intelligent Information Systems Conference, 2001.
3. J. Hansen and S. Patil. "Speech Under Stress: Analysis, Modeling, and Recognition." LNAI 4343:108-137, 2007.
4. J. Tao, Y. Kang, and A. Li. "Prosody Conversion from Neutral Speech to Emotional Speech." ICASSP, 2006.
5. M. Liberman, K. Davis, M. Grossman, N. Martey, and J. Bell. "Emotional Prosody Speech and Transcripts." Linguistic Data Consortium, 2002.
6. Z. Xiao, E. Dellandrea, W. Dou, and L. Chen. "Features Extraction and Selection for Emotional Speech Classification." IEEE, 2005.

Project Overview

Task Description

We seek to classify emotion of a human voice without regard to semantic content or situational context. We identified a database containing nearly 3 hours of high-quality data from the Linguistic Data Consortium. It contained labeled emotional snippets intoned by professional actors. We hope to outperform a human baseline in correctly choosing the emotion expressed in each phrase from a list of 15 categories.

Outline of Approach

This dataset is first parsed to select viable samples and preprocessed to remove bias. Each sample is then divided into segments and fed into an algorithm originally developed for speech compression, Linear Predictive Coding (LPC). We use the output of this filter to generate emotional features of each trial. Each of these feature vectors is then used to train a three-layer neural network. We evaluated our results on an independent test database.

Database

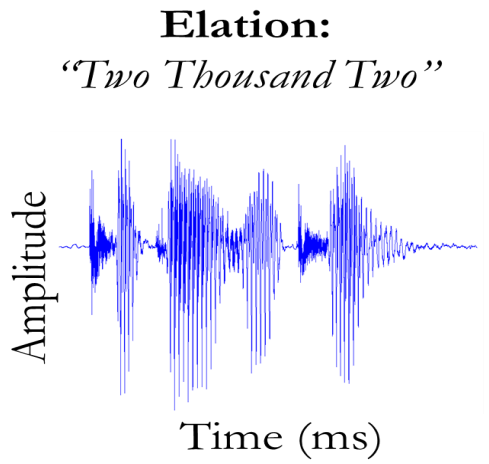
Emotion Database

We use the Emotional Prosody Speech and Transcripts Database from the Linguistic Data Consortium. The data set consists of labeled emotional utterances of semantically neutral content, intoned by professional actors.

Emotion Categories

Neutral
Disgust
Panic
Anxiety
Hot Anger
Cold Anger
Despair
Sadness
Elation
Happy
Interest
Boredom
Shame
Pride
Contempt

Specifically, the utterances were four-syllable dates and numbers, in the English language, from 15 emotion categories. Actors were provided with descriptions of each emotional context and were to reproduce the utterance in that manner. For example,



9 Actors participated, with a yield of ~3 hours of high quality speech samples. This translated to ~3000 speech samples overall, with ~200 samples per emotion. This was the largest English data set we could find with a high emotion resolution.

We split the database into two sets – a training set and test set, 70% - 30% respectively. These were used to train the neural network then test its performance.

Source

M. Liberman, K. Davis, M. Grossman, N. Martey, and J. Bell. "Emotional Prosody Speech and Transcripts." Linguistic Data Consortium, 2002.

Preprocessing

Database Parsing

This database was selected because it contained labeled emotional samples. The annotation files followed the general format

Start Stop A: Emotion,Phrase

Consequently, a parsing file was written that scanned each line and searched for 'A:'. If all sample descriptors could be identified successfully, the sample was added to the database providing that

- The start and stop times were less than 3 seconds apart
- The emotion matched one of the designated 15 categories
- The phrase did not contain any non-alpha characters (ie, [ ] or)

With the database assembled, the files were read individually and each sample extracted one at a time.

Silence Removal

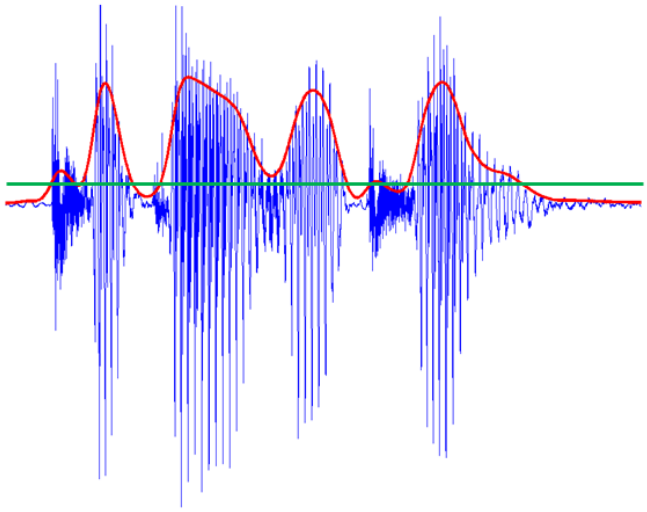
Although the annotations were always accurate, they were not exactly precise. The padding of silence that surrounded each sample added bias to our feature vectors. Consequently, we developed a method to detect and remove this padding. We want to apply a threshold above which we deem the speaker to be active. Directly applying such a threshold to the raw signal is risky because of the inherent background noise. Thus, we first find the amplitude envelope. The Hilbert transform was applied to each sample. This returns the analytic signal. A lowpass FIR filter with cutoff frequency 20 Hz was implemented in MatLab. Finally, we applied a zero-phase filtering technique. This essentially filters the signal in the forward direction and then again in the reverse direction. Because the phase shift of a FIR filter is linear in the passband, the phase shift in each direction is cancelled. This produces an excellent amplitude envelope with no offset in the time domain.

analyt = hilbert(trace);

b = fir1(200,20/(fs/2),'low');

env = filtfilt(b,1,abs(analyt));

Finally, we applied a generous threshold, found the first and last crossings of that threshold, and deleted the silence padding.



Linear Predictive Coding

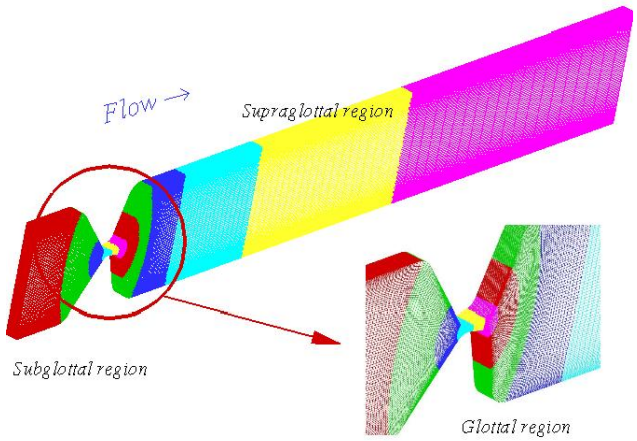
LPC Implementation

Linear Predictive Coding or LPC is usually applied for speech compression but we will be utilizing its very output dependent form for analysis of our signal. The fundamental principle behind LPC is that one can predict or approximate the future elements of a signal based on linear combinations of the previous signal elements and, of course, and an input excitation signal.

$$s(n) = \sum_{k=1}^{p} a_k s(n-k) + Gu(n)$$

All equations are from:
http://cs.haifa.ac.il/~nimrod/Compression/Speech/S4LinearPredictionCoding2009.pdf
.

The effectiveness of this model stems from fact that the processes that generate speech in the human body are relatively slow and that the human voice is quite limited in frequency range. The physical model of the vocal tract in LPC is a buzzer, corresponding the glottis which produces the baseline buzz in the human voice, at the end of a tube which has linear characteristics.

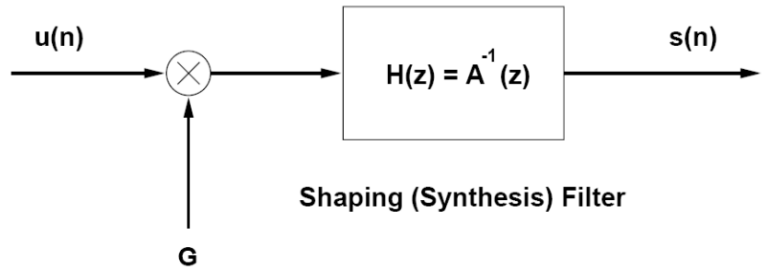


Courtesy of
https://engineering.purdue.edu/CFDLAB/projects/voice.htm
l

In “systems” speak we clearly see the form of a feedback filter emerging so to further analyze the system we take its Z-transform and find a transfer function from U[z] to S[z].

$$H(z) = \frac{S(z)}{GU(z)} = \frac{1}{1 - \Sigma_{k=1}^{p} a_k z^{-k}} = \frac{1}{A(z)}$$

The result is clearly an all pole filter and in standard application one would feed in the generating signal and get out a compressed version of the output.



The key barrier to implementing this filter is of course determining the “a” values or the coefficients of our linear superposition approximation of the output signal. Ultimately when we form the linear superposition, we want to choose coefficients that yield a compressed signal with the minimum deviation from the original signal; equivalently we want to minimize the difference(error) between the two signals.

$$e(n) = x(n) - \hat{x}(n)$$

From the form of s(n) we can derive and equivalent condition on the auto-correlation R[j].

$$\sum_{i=1}^{p} a_i R(j-i) = -R(j)$$

Where:

$$R(i) = E\{s(n)\, s(n-k)\}$$

Thus, we have p such equations, one for each R(j) and so we can more easily describe our conditions in terms of a matrix equation.

$$\begin{bmatrix} R_0 & R_1 & \cdots & R_{k-1} \\ R_1 & R_0 & \cdots & R_{k-2} \\ \vdots & \vdots & \ddots & \vdots \\ R_{k-1} & R_{k-2} & \cdots & R_0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_k \end{bmatrix} = - \begin{bmatrix} R_0 \\ R_1 \\ \vdots \\ R_{k-1} \end{bmatrix}$$

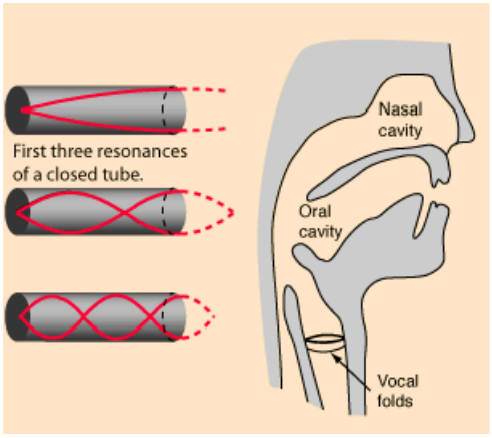
The matrix we now need to invert and multiply has a unique constant diagonal structure which classifies it as a Toeplitz matrix. There have been multiple methods developed for solving equations with Toeplitz matrices and one of the most efficient method, the method we used, is the the Levinson Durbin algorithm. This method is a bit involved but fundamentally it solves the system of equations by solving smaller sub-matrix equations and iterating up to get a recursive solution for all the coefficients.

Application

To reapply this method, this filter, towards our goal of speech analysis we first note that the form of the filter primarily dependent on the output rather than the input. The coefficients that we derived using the Levinson Durbin algorithm only use properties (the auto-correlation) of the output signal rather than the input signal. This means that this filter, in a way, is more natural as a method for going form output to input rather than the reverse, all we need do is take the reciprocal of the transfer function.

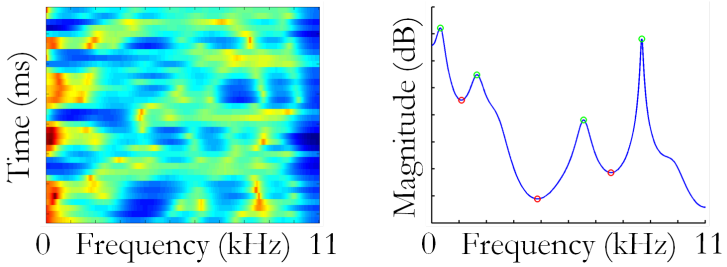
$$H(z)^{-1} = \frac{GU(z)}{S(z)} = 1 - \Sigma_{k=1}^{p} a_k z^{-k}$$

We go from an all pole filter to an all zero filter which now takes in a speech signal and returns the generating signal. This transfer function is actually more useful for our purposes because of our method of analyzing speech signals. We are primarily looking to identify the formants in the speech signal, the fundamental components of phonemes that make up human alphabets. These formants directly correspond to the resonant modes of the vocal tract, so we are effectively trying to achieve a natural mode decomposition of a complex resonant cavity.



Courtesy of http://hyperphysics.phy-astr.gsu.edu/hbase/music/vocres.html

Therefore these formants are more easily identifiable in the generating signal (since there are inherently a property of the generating cavity). With the filter generated by LPC we can now reconstruct a linear approximation of the generation signal using the speech signals from our soundbank. Our full signal of course can be represented by a spectrogram and the formant correspond to the local maxima of each time slice of the spectrogram.



Spectrogram (left); One slice of the spectrogram,

with peaks and troughs highlighted (right)

What we chose to extract from these spectrograms were the amplitude and frequency data of the first 4 formants present in the signal, as these are usually the most dominant, as well as the same information about the minima in between the peaks. This is the information we will need to feed into our classifier for emotion classification.

Feature Extraction

Formants

We have found the pitch contour, essentially the amplitude envelope of the Fourier transform. The peaks in this frequency response correspond to resonant modes of the human vocal tract. We want to analyze how the location and magnitude of these peaks relate to emotional content. Therefore, we need to find the peaks and valleys in each LPC slice of every sample.

Peak and Valley Detection

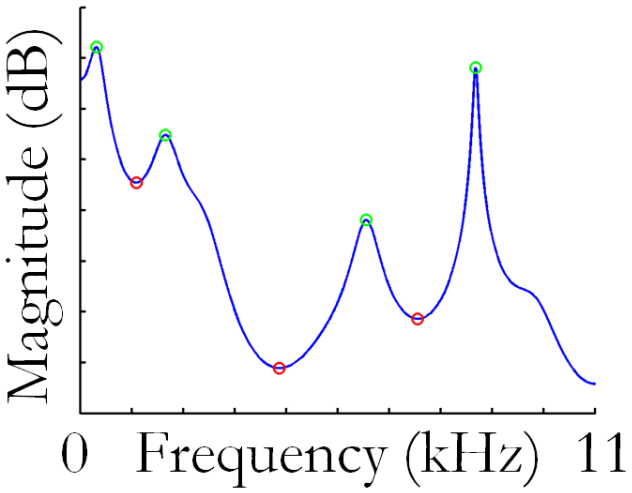
Because the LPC produces a smooth pitch contour, we employed a simple first derivative maximum and minimum test. Implemented with a for-loop, this proves to be exceedingly slow when even a short phrase is divided into about 40 time windows, each with several thousand frequency bins. We vectorized this operation with offset matrices and logical indexing.

Maximum
Equation:

$$\begin{aligned} contour(:, 1:end-2) &< contour(:, 2:end-1) \& \\ contour(:, 3:end) &< contour(:, 2:end-1) \end{aligned}$$

Minimum
Equation:

$$\begin{aligned} contour(:, 1:end-2) &> contour(:, 2:end-1) \& \\ contour(:, 3:end) &> contour(:, 2:end-1) \end{aligned}$$



Feature Vector

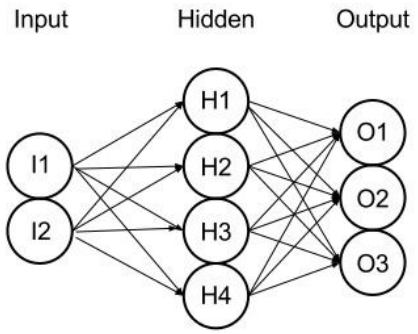
We grabbed the frequency location and magnitude of the first four formants and first three valleys (more than 90 percent of pitch contours contained at least this number of formants). We also calculated the mean and variance of power in the sample from the L2 norm. Finally, since all phrases were four syllables long, we used length as a measure of speed. These features were all passed to the neural network.

Neural Networks

Neural Network Implementation

For our classification task, we implement a multiclass artificial neural network. Artificial neural networks are computing structures that attempt to mimic the brain. They perform by means of distributed processing among nodes, which correspond to neurons. Connection strength between individual neurons collectively determines the network's activity, similar to biological synapse performance. Due to the computational flexibility inherent in these large, distributed systems, neural networks excel at nonlinear tasks such as facial or speech recognition. We expected that a neural network would excel at our task: emotion classification.

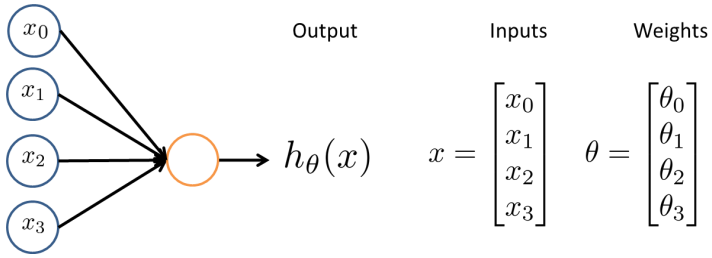
Network Specifications



Example of a multilayer perceptron.

We utilize the common multilayer perceptron neural network with neuron activation modeled by a sigmoid.

Neuron Model



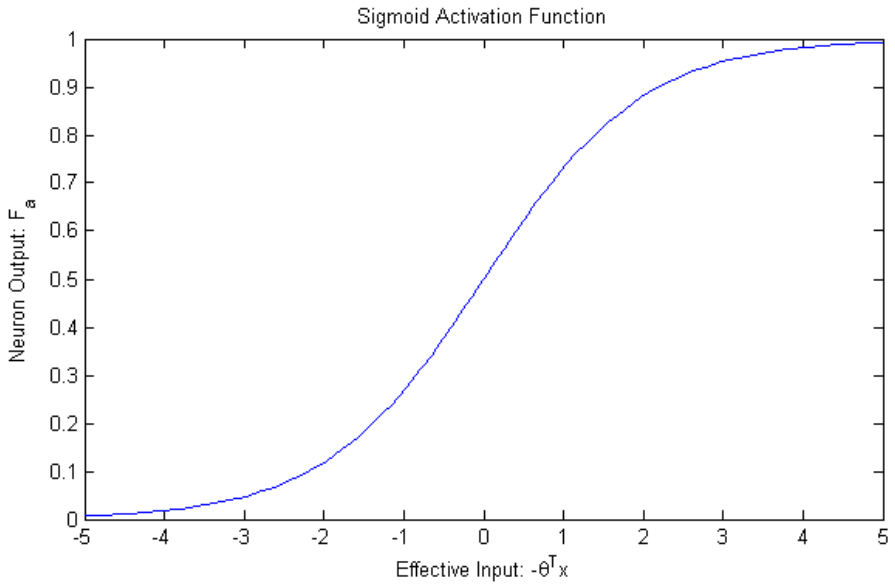
Courtesy of coursera.org

$$h_\theta = F_a\left[\sum_i x_i\theta_i\right]$$

Neurons are heavily connected – one connects to many, and many to others. While each neuron takes in multiple inputs, each input is not weighted equally. The weight determines how strongly an input will affect a neuron's output, hθ. The activation function Fa determines the relation between the inputs and the output.

The neurons of our network are characterized by the sigmoid activation function.

$$F_a = \frac{1}{1 + e^{-\theta^T x}}$$



The sigmoid neuron has two distinct advantages. First, the sigmoid activation function is differentiable. This simplifies the function used in calculating cost for training. Second, the sigmoid activation function retains thresholding – output tends toward either 1 or 0. This enables the neurons to behave as ‘on/off’ nonlinear switches.

Network Architecture

We use the same type of network as displayed in fig. 1: a multilayer perceptron. This network is feed forward, fully connected, and contains >= three layers. Feed forward means that signals travel straight through the network, from input to output, via hidden layers. No connections exist between neurons of the same layer. Fully connected means that each neuron of a previous layer has a connection between every neuron of the next layer. Feed forward systems have comparatively manageable training mechanisms, and so are preferred to other connection systems.

The input layer of a multiperceptron takes in one feature per node. The output layer contains one class per node. This layer yields the probability that an input is that particular class. The hidden layers perform operations on the inputs to yield the output. The hidden layers are thought to flesh out

the distinct features that relate input to output, though what they flesh out is difficult to obtain as well as to understand.

For our application, speech features such as formant locations and power statistics are fed into 14 input nodes, and the emotion category is extracted from the 15 output nodes. The number of input nodes depends on the number of speech features, 14, that we are classifying by. The number of output nodes corresponds to the 15 emotion classes samples are categorized into. In between this input and output layer lies one hidden layer of approximately 100 artificial neurons. Our network necessitated such a large number of neurons due to the complex relation between speech features and emotion.

Training

To tailor the network for our classification task, we train its weights via feedforward-back propagation, a popular means to minimize a neural network's cost function: J(Θ). The process is as follows:

- Randomly initialize weights between nodes. We take Θ = weight matrix.
- Implement forward propagation to determine J(Θ).

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log(h_\Theta(x^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - (h_\Theta(x^{(i)}))_k)\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{ji}^{(l)})^2$$

Courtesy of coursera.org

In forward propagation, all training samples are fed into the initialized neural network to obtain estimated outputs, which are used to determine J(Θ). To prevent overfitting, the cost function includes a regularization term with λ = 1, which enables control over the overall magnitudes of Θ parameters.

- Implement back propagation to determine the gradient of J(Θ). In back propagation, the error derivatives of latter layer weights are used to calculate error derivatives of former layer weights, up until all weight derivatives are found.
- Use an optimization algorithm (in our case, batch gradient descent) to minimize J(Θ), and obtain a the correct set of weights. To increase the speed of gradient descent, implement feature scaling, wherein each input feature is first shifted by the mean, and then scaled by the standard deviation.

$$x_{scaled} = \frac{x - \mu}{\sigma}$$

Neural networks tend to have multiple local optima. To determine the global optima, the back propagation algorithm is run through multiple iterations (we use 1000).

Performance Testing

We evaluate an independent test database with the trained neural network to test the network's performance.

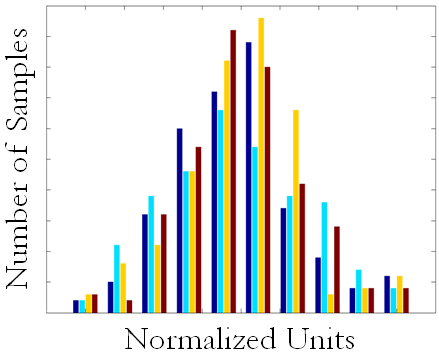
Useful References

To learn more about neural networks, check out Andrew Ng's Machine Learning or Geoffrey Hinton's Neural Networks for Machine learning, as found on coursera.org. Our neural network code is modified from Andrew Ng's course code.

Results

Feature Separation

The neural network must be fed meaningful information. If there are not clusters within the data and distinguishing relationships between features, than there is nothing for the neural network to learn. Here we show that the features extracted from the LPC are in fact providing useful information to the neural network. A subset of four emotions has been selected for visual clarity. They have been normalized by mean and variance to generate a roughly Gaussian distribution of the frequency location of the fourth formant. Notice that the mean, variance, and skewness of each of the four probability density functions is are distinct.



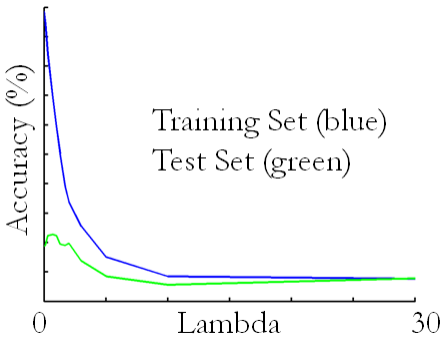
Computation Complexity

This algorithmic pipeline involves a significant number of computationally heavy tasks. In its original iteration, each sample required 50 seconds of processing time before the neural network was trained. We minimized the order of the filter to introduce an acceptable level of passband ripple, vectorized the LPC windowing implementation, and replaced the find operation used by in formant detection with logical indexing. These measures reduced the average computational time to only about half a second per sample.

Task	Database	Preprocess	LPC	Features	Total
Time (ms)	33.27	47.45	427.32	39.01	547.05

Neural Network Learning Curves

To evaluate the performance of the neural network, we split the database into two distinct sections. The training set (70% of the database) was used to train the neural network. A logarithmic distribution of regularization was tested. When each trained set of internodal weights was applied to the test set (30% of the database), we were able to determine the true accuracy of the neural network. We found that the test set accuracy actually peaked at a low value of regularization and then decreased. This is a characteristic signature of having far too small of a database. Given that the size of the database was fixed, we attempted to compensate by minimizing the number of input features to the neural network. It is important to have the ratio between training examples and inputs be large. If we had instead seen a gap between the training and test set performance even at high levels of regularization, we would have known that our choice of input features was inherently faulty.



Algorithm Performance

We established a human performance baseline by playing a randomized list of 100 samples to three individuals. None were able to significantly outperform chance. Clearly humans struggle to classify a precise emotion

from such a large list, especially without any context. If the test is altered to be a binary matching test where the subject is asked to evaluate whether a given emotion is intoned in a given sample, human performance markedly improves. This suggests that in normal everyday interaction, we rely on semantic content and situational context to derive an emotional expectation; then, we simply evaluate the speaker's tone to see if it matches that expectation.

The algorithm correctly identifies the tone of the speakers voice with 26 percent accuracy. The algorithm was particularly adept at detecting contempt (41%), anxiety (37%), sadness (33%), and boredom (31%). It struggled to identify cold anger (4%).

Subject	Chance	Human A	Human B	Human C	**Algorithm**
Accuracy	6.67	8.88	9.23	7.34	**26.14**

Future Directions

A neural network driven by features derived from the pitch contour of a speech sample is capable of robust emotional classification. The results presented here will be markedly improved with the expansion of existing short phrase, semantically neutral databases. These methods have been prepared for such an implementation through rigorous optimization of the computation complexity required to process each sample. A larger number of labeled samples will also allow for an enhanced feature vector of greater length. This guarantees an increase in the quality of classification. Independent and reliable emotional analysis adds rich texture to existing speech recognition methods that attempt to identify and decode each spoken word by allowing a machine to distinguish the intent of semantically identical phrases.

The Team

Abeer Javed: ECE!

John Cocjin: BIOE!

Kiefer Forseth: ECE, Music (Piano)!

Tapash J Sarkar: ECE, Math, Physics!

Acknowledgements

We would like to thank Professor Rich Baraniuk for his indefatigable energy, Christoph Studer for his advice, and Eva for delivering us the gift of MatLab this Christmas.

Cute Cat. Courtesy of Google Search on "Cute Cat"